UNIT13: STATIC MESH - GEOMETRY

【学習要項】
□OBJ File format
□OBJ File parser

【演習手順】
1．シェーダーファイルの追加（static_mesh.hlsli, static_mesh_vs.hlsl, static_mesh_ps.hlsl）
※内容は geometric_primitive と同じにする（インクルードするヘッダファイル名は変更する）

2．OBJ ファイル（.¥¥ resources¥¥cube.obj）をロードし、描画するクラス（static_mesh）を実装する
①プロジェクトに static_mesh.h と static_mesh.cpp を追加する
② static_mesh.h に static_mesh クラスを定義する（geometric_primitive.h からコピーして変更する）
※必要に応じてヘッダファイルをインクルードする

```
* 1:  class static_mesh
  2:  {
  3:  public:
  4:    struct vertex
  5:    {
  6:      DirectX::XMFLOAT3 position;
  7:      DirectX::XMFLOAT3 normal;
  8:    };
  9:    struct constants
 10:    {
 11:      DirectX::XMFLOAT4X4 world;
 12:      DirectX::XMFLOAT4 material_color;
 13:    };
 14:
 15:  private:
 16:    Microsoft::WRL::ComPtr<ID3D11Buffer> vertex_buffer;
 17:    Microsoft::WRL::ComPtr<ID3D11Buffer> index_buffer;
 18:
 19:    Microsoft::WRL::ComPtr<ID3D11VertexShader> vertex_shader;
 20:    Microsoft::WRL::ComPtr<ID3D11PixelShader> pixel_shader;
 21:    Microsoft::WRL::ComPtr<ID3D11InputLayout> input_layout;
 22:    Microsoft::WRL::ComPtr<ID3D11Buffer> constant_buffer;
 23:
 24:  public:
*25:    static_mesh(ID3D11Device* device, const wchar_t* obj_filename);
*26:    virtual ~static_mesh() = default;
 27:
 28:    void render(ID3D11DeviceContext* immediate_context,
 29:      const DirectX::XMFLOAT4X4& world, const DirectX::XMFLOAT4& material_color);
 30:
 31:  protected:
 32:    void create_com_buffers(ID3D11Device* device, vertex* vertices, size_t vertex_count,
 33:      uint32_t* indices, size_t index_count);
 34:  };
```

③ static_mesh.cpp に static_mesh コンストラクタを実装する（OBJ ファイルパーサーの実装）
※”cube.obj”をテキストエディタで開いて内容を確認する
※頂点と法線を読み込む（OBJ ファイルの仕様書（Obj Specification.txt）を参照）
※必要に応じてヘッダファイルをインクルードする

```
 1:  using namespace DirectX;
 2:  static_mesh::static_mesh(ID3D11Device* device, const wchar_t* obj_filename)
 3:  {
 4:      std::vector<vertex> vertices;
 5:      std::vector<uint32_t> indices;
 6:      uint32_t current_index{ 0 };
 7:
 8:      std::vector<XMFLOAT3> positions;
 9:      std::vector<XMFLOAT3> normals;
10:
11:      std::wifstream fin(obj_filename);
12:      _ASSERT_EXPR(fin, L"'OBJ file not found.");
13:      wchar_t command[256];
14:      while (fin)
```

```
    {
        fin >> command;
        if (0 == wcscmp(command, L"v"))
        {
            float x, y, z;
            fin >> x >> y >> z;
            positions.push_back({ x, y, z });
            fin.ignore(1024, L'¥n');
        }
        else if (0 == wcscmp(command, L"vn"))
        {
            FLOAT i, j, k;
            fin >> i >> j >> k;
            normals.push_back({ i, j, k });
            fin.ignore(1024, L'¥n');
        }
        else if (0 == wcscmp(command, L"f"))
        {
            for (size_t i = 0; i < 3; i++)
            {
                vertex vertex;
                size_t v, vt, vn;

                fin >> v;
                vertex.position = positions.at(v - 1);
                if (L'/' == fin.peek())
                {
                    fin.ignore(1);
                    if (L'/' != fin.peek())
                    {
                        fin >> vt;
                    }
                    if (L'/' == fin.peek())
                    {
                        fin.ignore(1);
                        fin >> vn;
                        vertex.normal = normals.at(vn - 1);
                    }
                }
                vertices.push_back(vertex);
                indices.push_back(current_index++);
            }
            fin.ignore(1024, L'¥n');
        }
        else
        {
            fin.ignore(1024, L'¥n');
        }
    }
    fin.close();

    create_com_buffers(device, vertices.data(), vertices.size(), indices.data(), indices.size());

        :
        : 省略（シェーダーオブジェクトの生成）
        :

}
```

④ static_mesh.cpp に static_mesh クラスの `create_com_buffers` メンバ関数を実装する
※内容は `geometric_primitive` クラスのものと同じにする

⑤ static_mesh.cpp に static_mesh クラスの render メンバ関数を実装する
※内容は `geometric_primitive` クラスのものと同じにする

3．framework クラスのメンバ変数として static_mesh *型配列を要素数 8 で宣言する

```
std::unique_ptr<static_mesh> static_meshes[8];
```

4．framework クラスの initialize メンバ関数で static_mesh オブジェクトを生成する

```
static_meshes[0] = std::make_unique<static_mesh>(device.Get(), L".¥¥resources¥¥cube.obj");
```

5．framework クラスの render メンバ関数で static_mesh クラスの render メンバ関数を呼び出す
①拡大縮小（S）・回転（R）・平行移動（T）行列を計算する
②上記 3 行列を合成しワールド変換行列（world）を作成する
③static_mesh クラスの render メンバ関数を呼び出す

```
static_meshes[0]->render(immediate_context.Get(), world, { 0.5f, 0.8f, 0.2f, 1.0f });
```

6．実行し、正立方体が描画される事を確認する

7．実装完了後、別の OBJ ファイル（torus.obj）をロードして、描画されることを確認する

【評価項目】
□OBJ ファイルのロード・描画